Homework 3 code

1. Loan calculations

```
public class LoanCalc {

static double epsilon = 0.001;  // The computation tolerance (estimation error)
static int iterationCounter;    // Monitors the efficiency of the calculation

public static void main(String[] args) {
// Gets the loan data
double loan = Double.parseDouble(args[0]);
double rate = Double.parseDouble(args[1]);
int n = Integer.parseInt(args[2]);
System.out.println("Loan sum = " + loan + ", interest rate = " + rate + "%, periods = "
+ n);

// Computes the periodical payment using brute force search
System.out.print("Periodical payment, using brute force: ");
System.out.printf("%.2f", bruteForceSolver(loan, rate, n, epsilon));
System.out.println();
System.out.println("number of iterations: " + iterationCounter);

// Computes the periodical payment using bisection search
System.out.print("Periodical payment, using bi-section search: ");
System.out.printf("%.2f", bisectionSolver(loan, rate, n, epsilon));
System.out.println();
System.out.println("number of iterations: " + iterationCounter);
        }
```

```
public static double bruteForceSolver(double loan, double rate, int n, double epsilon)
{
 iterationCounter = 0;
double g = loan / n;
while (endBalance(loan, rate, n, g) > epsilon) {
          g += epsilon;  // Increase the guess
          iterationCounter++;
        }

 return g;
    }

public static double bisectionSolver(double loan, double rate, int n, double epsilon) {
          iterationCounter = 0;
double L = 0.0;
double H = loan;
double g = (L + H)/2;
double fL = endBalance(loan, rate, n, L);
double fH = endBalance(loan, rate, n, H);
double fg = endBalance(loan, rate, n, g);

while ((H-L) > epsilon) {
       if ((fg * fL) > 0) {
            L = g;
            fL = fg;
  }   else {
           H = g;
           fH = fg;
 }
      g = (L + H)/2;
      fg = endBalance(loan, rate, n, g);
```

```
        iterationCounter++;
}

          return g;
}


private static double endBalance(double loan, double rate, int n, double payment) {
    double x = loan;
    double rateIn = rate/100;
     for (int i = 0; i < n; i++) {
          x -= payment;
          x += (rateIn * x);
          }
          return x;
     }
}
```

2. Lower case

```
public class LowerCase {
    public static void main(String[] args) {
        String str = args[0];
        System.out.println(lowerCase(str));
    }

   /**
    * Returns a string which is identical to the original string,
    * except that all the upper-case letters are converted to lower-case letters.
    * Non-letter characters are left as is.
    */
    public static String lowerCase(String str) {
    int i=0;
    String answer = "";
    while (i< str.length()) {
        char currentChar= str.charAt(i);

        if (Character.isUpperCase(currentChar)) {
            answer+= (char) (currentChar+32);
        } else {
            answer+= currentChar;
        }

        i++;
    }
        return answer;
    }
}
```

3. Unique characters

```
public class UniqueChars {
    public static void main(String[] args) {
        String str = args[0];
        System.out.println(uniqueChars(str));
    }

    /**
     * Returns a string which is identical to the original string,
     * except that all the duplicate characters are removed,
     * unless they are space characters.
     */
    public static String uniqueChars(String str) {
       String newString= " ";
       int i= 0;

       while(i<str.length()) {
        char currentChar= str.charAt(i);

    if (currentChar != ' ' && newString.indexOf(String.valueOf(currentChar)) == -1) {
          newString += currentChar;
     } else if (currentChar == ' ') {
        newString+=' ';
     }
     i++;
}
        return newString.trim();
    }
}
```

4. Calendar

```
public class Calendar {
    static int dayOfMonth = 1;
    static int month = 1;
    static int year = 1900;
    static int dayOfWeek = 2;     // 1.1.1900 was a Monday

    public static void main(String args[]) {

        int givenYear = Integer.parseInt(args[0]);

        while (year < givenYear) {
          advance();
        }



        while (year <= givenYear && month <= 12) {
            if (dayOfWeek == 1) {
                System.out.println(dayOfMonth + "/" + month + "/" + year + " Sunday");
            } else {
                System.out.println(dayOfMonth + "/" + month + "/" + year);
            }
            advance();
        }
    }

    private static void advance() {
        dayOfWeek = (dayOfWeek % 7) + 1;
        dayOfMonth++;
```

```
        if (dayOfMonth > nDaysInMonth(month, year)) {
            dayOfMonth = 1;
            month++;

            if (month > 12) {
                month = 1;
                year++;
            }
        }
    }

    private static boolean isLeapYear(int year) {
        return (year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0));
    }

    public static int nDaysInMonth(int month, int year) {
        switch (month) {
            case 2:
                return isLeapYear(year) ? 29 : 28;
            case 4:
            case 6:
            case 9:
            case 11:
                return 30;
            default:
                return 31;
        }
    }
}
```